```
import numpy as np
import pandas as pd
```

```
data = pd.read_csv("auto-mpg.csv")
```

```
data
```

	mpg	cylinders	displacement	horsepower	weight	acceleration	model year	origin	car name
0	18.0	8	307.0	130	3504	12.0	70	1	chevrolet chevelle malibu
1	15.0	8	350.0	165	3693	11.5	70	1	buick skylark 320
2	18.0	8	318.0	150	3436	11.0	70	1	plymouth satellite
3	16.0	8	304.0	150	3433	12.0	70	1	amc rebel sst
4	17.0	8	302.0	140	3449	10.5	70	1	ford torino
...	...	...	...	...	...	...	...	...	...
393	27.0	4	140.0	86	2790	15.6	82	1	ford mustang gl
394	44.0	4	97.0	52	2130	24.6	82	2	vw pickup
395	32.0	4	135.0	84	2295	11.6	82	1	dodge rampage
396	28.0	4	120.0	79	2625	18.6	82	1	ford ranger
397	31.0	4	119.0	82	2720	19.4	82	1	chevy s-10

398 rows × 9 columns

```
mpg = data["mpg"].to_numpy()    #The mean, median, and standard deviation of mpg
```

```
print(np.mean(mpg))
print(np.median(mpg))
print(np.std(mpg))
```

```
23.514572864321607
23.0
7.806159061274433
```

```
mpg_above_25 = print(np.sum(mpg > 25))        #The number of cars with mpg greater than 25
```

```
158
```

```
car_names = print(data[np.array(data["cylinders"]) > 6]["car name"].tolist()) #Using NumPy, filter all cars with more than 6 cylinders in lis
```

```
['chevrolet chevelle malibu', 'buick skylark 320', 'plymouth satellite', 'amc rebel sst', 'ford torino', 'ford galaxie 500', 'chevrolet
```

```
percentiles = print(np.percentile(data["weight"], [25, 50, 75])) #Compute the 25th, 50th, and 75th percentiles of the weight column using Num
```

```
[2223.75 2803.5  3608.  ]
```

```
acceleration_array = data["acceleration"].to_numpy() #convert the acceleration column into a NumPy array and normalize its values
```

```
normalized_acceleration = print((acceleration_array - np.min(acceleration_array)) / (np.max(acceleration_array) - np.min(acceleration_array))
```

```
[0.23809524 0.20833333 0.17857143 0.23809524 0.14880952 0.11904762
 0.05952381 0.0297619  0.11904762 0.0297619  0.11904762 0.
 0.08928571 0.11904762 0.41666667 0.44642857 0.44642857 0.47619048
 0.38690476 0.74404762 0.56547619 0.38690476 0.56547619 0.26785714
 0.41666667 0.35714286 0.41666667 0.32738095 0.625      0.38690476
 0.44642857 0.35714286 0.6547619  0.29761905 0.44642857 0.44642857
 0.44642857 0.44642857 0.23809524 0.20833333 0.32738095 0.29761905
 0.20833333 0.23809524 0.23809524 0.32738095 0.6547619  0.41666667
 0.38690476 0.35714286 0.35714286 0.68452381 0.38690476 0.6547619
 0.5952381  0.6547619  0.74404762 0.44642857 0.53571429 0.92261905
 0.68452381 0.50595238 0.23809524 0.23809524 0.32738095 0.29761905
 0.20833333 0.17857143 0.32738095 0.32738095 0.26785714 0.32738095
 0.26785714 0.35714286 0.47619048 0.35714286 0.38690476 0.5952381
 0.68452381 0.5952381  0.47619048 0.53571429 0.38690476 0.41666667
```

```
0.50595238 0.29761905 0.20833333 0.29761905 0.38690476 0.26785714
0.20833333 0.23809524 0.29761905 0.38690476 0.17857143 0.17857143
0.17857143 0.50595238 0.5952381  0.47619048 0.50595238 0.47619048
0.77380952 0.35714286 0.26785714 0.29761905 0.26785714 0.41666667
0.6547619  0.68452381 0.50595238 0.32738095 0.625      0.35714286
0.44642857 0.29761905 0.08928571 0.68452381 0.44642857 0.35714286
0.44642857 0.17857143 0.35714286 0.32738095 0.17857143 0.50595238
0.53571429 0.47619048 0.53571429 0.6547619  0.50595238 0.77380952
0.53571429 0.53571429 0.5952381  0.50595238 0.35714286 0.38690476
0.32738095 0.47619048 0.44642857 0.50595238 0.44642857 0.38690476
0.50595238 0.6547619  0.38690476 0.44642857 0.35714286 0.41666667
0.44642857 0.47619048 0.47619048 0.47619048 0.77380952 0.68452381
0.20833333 0.35714286 0.38690476 0.32738095 0.77380952 0.625
0.6547619  0.6547619  0.41666667 0.32738095 0.23809524 0.47619048
0.53571429 0.47619048 0.625      0.32738095 0.50595238 0.53571429
0.38690476 0.35714286 0.53571429 0.41666667 0.53571429 0.38690476
0.32738095 0.56547619 0.44642857 0.5297619  0.41071429 0.57738095
0.43452381 0.29761905 0.29761905 0.35119048 0.28571429 0.44047619
0.38690476 0.57142857 0.57142857 0.8452381  0.83928571 0.36904762
0.55952381 0.57738095 0.77380952 0.48809524 0.58333333 0.25
0.53571429 0.5        0.33333333 0.45833333 0.30952381 0.82738095
0.44642857 0.51785714 0.24404762 0.23809524 0.41666667 0.35714286
0.625      0.4047619  0.63095238 0.44642857 0.52380952 0.26785714
0.6547619  0.33928571 0.41071429 0.5        0.5297619  0.57738095
0.6547619  0.18452381 0.20238095 0.25       0.38690476 0.38690476
0.47619048 0.60714286 0.46428571 0.53571429 0.4702381  0.5
0.36309524 0.38690476 0.28571429 0.32738095 0.80357143 0.38095238
0.67857143 0.63095238 0.5        0.44642857 0.30952381 0.28571429
0.66666667 0.60714286 0.46428571 0.44047619 0.54761905 0.54761905
0.46428571 0.51785714 0.63690476 0.42261905 0.30952381 0.32142857
0.19047619 0.33928571 0.50595238 0.36904762 0.39880952 0.38690476
0.4047619  0.51785714 0.57142857 0.41071429 0.4702381  0.33333333
0.45833333 0.46428571 0.41071429 0.51190476 0.44047619 0.60714286
0.55357143 0.60714286 0.51190476 0.44047619 0.32142857 0.30952381
0.42857143 0.41071429 0.375      0.41666667 0.29761905 0.35714286
0.42857143 0.38095238 0.41666667 0.7202381  0.55952381 1.
0.8452381  0.30952381 0.41071429 0.66666667 0.39880952 0.47619048
0.19642857 0.29166667 0.30952381 0.39880952 0.64285714 0.44642857
0.5        0.50595238 0.60119048 0.7202381  0.63690476 0.46428571
0.44642857 0.56547619 0.41666667 0.42857143 0.58928571 0.38095238
0.66666667 0.81547619 0.93452381 0.70833333 0.82142857 0.3452381
0.55357143 0.5952381  0.43452381 0.20238095 0.26785714 0.42261905
0.375      0.53571429 0.45833333 0.5        0.38095238 0.27380952
0.29166667 0.5297619  0.5        0.48214286 0.58333333 0.67857143
```

```
data["horsepower"] = pd.to_numeric(data["horsepower"], errors="coerce") #Increase all horsepower values by 10% and store the updated values
```

```
mean_horsepower = data["horsepower"].mean()
data["horsepower"] = data["horsepower"].fillna(mean_horsepower)
```

```
horsepower_array = data["horsepower"].to_numpy()
```

```
updated_horsepower = print(horsepower_array * 1.10)
```

```
103.4         99.          93.5        117.7         99.
159.5        253.          53.9         82.5        100.1
123.2        165.         121.         134.2        198.
104.5        114.91632653 110.         110.          73.7
```

```
 85.8        106.7        121.         121.         52.8
 72.6         57.2         77.          66.        121.
154.         152.9        115.5        104.5         93.5
 96.8        110.          99.         115.5         93.5
121.         132.         159.5        181.5        152.9
154.          74.8        104.5        106.7         82.5
104.5        115.5         93.5        106.7        113.3
137.5        126.5        146.3         78.1         74.8
126.5         93.5         96.8         99.        121.
143.         141.9        151.8        148.5        170.5
156.2        137.5        165.          78.1         71.5
 88.          88.          84.7        137.5         78.1
 99.          77.          77.          71.5         75.9
 99.         126.5        126.5         99.          83.6
 66.          77.          71.5         99.          96.8
 99.          99.          85.8         99.          82.5
101.2         82.5         71.5        115.5         71.5
 52.8         52.8         73.7         73.7         73.7
114.91632653  73.7         68.2        145.2        110.
 96.8        114.91632653  79.2         92.4         92.4
101.2        121.          92.4         63.8         70.4
 66.          73.7         71.5         68.2         74.8
 69.3         71.5         71.5         81.4        114.91632653
 82.5         82.5        110.          81.4         88.
 83.6        127.6        132.         121.         115.5
 96.8         93.5         96.8         96.8         96.8
 93.5         92.4         99.         101.2        114.91632653
 81.4         74.8         74.8         69.3         77.
 96.8         82.5         77.          73.7         73.7
 73.7        121.          93.5        101.2        123.2
105.6         92.4         99.          94.6         57.2
 92.4         86.9         90.2       ]
```

```
data["origin"] = pd.to_numeric(data["origin"], errors="coerce") #Find the average displacement of cars with an origin of 2 (Europe) using Num
```

```
displacement_array = data["displacement"].to_numpy()
origin_array = data["origin"].to_numpy()
```

```
average_displacement_europe = print(np.mean(displacement_array[origin_array == 2]))
```

```
109.14285714285714
```

```
mean_horsepower = data["horsepower"].mean() #Create a 2D NumPy array containing the columns mpg, horsepower, and weight.Compute the dot prod
data["horsepower"] = data["horsepower"].fillna(mean_horsepower)
```

```
matrix = data[["mpg", "horsepower", "weight"]].to_numpy()
```

```
vector = np.array([1, 0.5, -0.2])
```

```
dot_product_result = np.dot(matrix, vector)
```

```
print(dot_product_result[:10])
```

```
[-617.8 -641.1 -594.2 -595.6 -602.8 -754.2 -746.8 -740.9 -758.5 -660. ]
```

```
model_year_array = data["model year"].to_numpy() #Use NumPy to sort the cars by model_year in descending order and display the firstfive car
```

```
sorted = np.argsort(-model_year_array)
```

```
sorted_car_names = print(data["car name"].to_numpy()[sorted][:5])
```

```
['chevy s-10' 'ford ranger' 'dodge rampage' 'vw pickup' 'ford mustang gl']
```

```
mpg = data["mpg"].values #Compute the Pearson correlation coefficient between mpg and weight usingNumPy.
weight = data["weight"].values
```

```
correlation_coefficient = print(np.corrcoef(mpg, weight)[0, 1])
```

```
-0.8317409332443352
```

```
mean_mpg_per_cylinder = print(data.groupby("cylinders")["mpg"].mean().to_dict()) #Calculate the mean mpg for cars grouped by the number of c
```

```
{3: 20.55, 4: 29.28676470588235, 5: 27.366666666666664, 6: 19.985714285714284, 8: 14.963106796116506}
```

```
import pandas as pd
```

```
df = pd.read_csv("auto-mpg.csv")
```

```
df
```

	mpg	cylinders	displacement	horsepower	weight	acceleration	model year	origin	car name
0	18.0	8	307.0	130	3504	12.0	70	1	chevrolet chevelle malibu
1	15.0	8	350.0	165	3693	11.5	70	1	buick skylark 320
2	18.0	8	318.0	150	3436	11.0	70	1	plymouth satellite
3	16.0	8	304.0	150	3433	12.0	70	1	amc rebel sst
4	17.0	8	302.0	140	3449	10.5	70	1	ford torino
...	...	...	...	...	...	...	...	...	...
393	27.0	4	140.0	86	2790	15.6	82	1	ford mustang gl
394	44.0	4	97.0	52	2130	24.6	82	2	vw pickup
395	32.0	4	135.0	84	2295	11.6	82	1	dodge rampage
396	28.0	4	120.0	79	2625	18.6	82	1	ford ranger
397	31.0	4	119.0	82	2720	19.4	82	1	chevy s-10

398 rows × 9 columns

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
print(df.head(10)) #Load the dataset into a Pandas DataFrame. Display: The first 10 rows The total number of rows and columns Summary statis
```

```
    mpg  cylinders  displacement horsepower  weight  acceleration  model year  \
0  18.0          8         307.0        130    3504          12.0          70
1  15.0          8         350.0        165    3693          11.5          70
2  18.0          8         318.0        150    3436          11.0          70
3  16.0          8         304.0        150    3433          12.0          70
4  17.0          8         302.0        140    3449          10.5          70
5  15.0          8         429.0        198    4341          10.0          70
6  14.0          8         454.0        220    4354           9.0          70
7  14.0          8         440.0        215    4312           8.5          70
8  14.0          8         455.0        225    4425          10.0          70
9  15.0          8         390.0        190    3850           8.5          70

   origin                   car name
0       1  chevrolet chevelle malibu
1       1          buick skylark 320
2       1         plymouth satellite
3       1              amc rebel sst
4       1                ford torino
5       1           ford galaxie 500
6       1           chevrolet impala
7       1          plymouth fury iii
8       1           pontiac catalina
9       1          amc ambassador dpl
```

```
print("Rows and Columns:", df.shape)
```

```
Rows and Columns: (398, 9)
```

```
print(df.describe())
```

```
              mpg   cylinders  displacement       weight  acceleration  \
count  398.000000  398.000000    398.000000   398.000000    398.000000
mean    23.514573    5.454774    193.425879  2970.424623     15.568090
std      7.815984    1.701004    104.269838   846.841774      2.757689
min      9.000000    3.000000     68.000000  1613.000000      8.000000
25%     17.500000    4.000000    104.250000  2223.750000     13.825000
50%     23.000000    4.000000    148.500000  2803.500000     15.500000
75%     29.000000    8.000000    262.000000  3608.000000     17.175000
max     46.600000    8.000000    455.000000  5140.000000     24.800000
```

```
       model year      origin
count  398.000000  398.000000
mean    76.010050    1.572864
std      3.697627    0.802055
min     70.000000    1.000000
25%     73.000000    1.000000
50%     76.000000    1.000000
75%     79.000000    2.000000
max     82.000000    3.000000
```

```
df_filtered = print(df[(df['model year'] == 75) & (df['weight'] < 3000)][['car name', 'weight', 'mpg']]) #Find all cars manufactured in 1975
```

```
              car name  weight   mpg
167     toyota corolla    2171  29.0
168         ford pinto    2639  23.0
169        amc gremlin    2914  20.0
170      pontiac astro    2592  23.0
171      toyota corona    2702  24.0
172  volkswagen dasher    2223  25.0
173         datsun 710    2545  24.0
174         ford pinto    2984  18.0
175  volkswagen rabbit    1937  29.0
177         audi 100ls    2694  23.0
178        peugeot 504    2957  23.0
179        volvo 244dl    2945  22.0
180          saab 99le    2671  25.0
181   honda civic cvcc    1795  33.0
```

```
print(df.isnull().sum()) #Identify if there are any missing values in the dataset. Replace missing values in thehorsepower column with the c
```

```
mpg             0
cylinders       0
displacement    0
horsepower      0
weight          0
acceleration    0
model year      0
origin          0
car name        0
dtype: int64
```

```
df['horsepower'] = pd.to_numeric(df['horsepower'], errors='coerce')
```

```
median_hp = df['horsepower'].median()
df['horsepower'] = df['horsepower'].fillna(median_hp)
```

```
print(df['horsepower'].isnull().sum())
```

```
0
```

```
df['power_to_weight_ratio'] = df['horsepower'] / df['weight'] #Add a new column power_to_weight_ratio, calculated as horsepower / weight
```

```
df.head()
```

	mpg	cylinders	displacement	horsepower	weight	acceleration	model year	origin	car name	power_to_weight_ratio
0	18.0	8	307.0	130.0	3504	12.0	70	1	chevrolet chevelle malibu	0.037100
1	15.0	8	350.0	165.0	3693	11.5	70	1	buick skylark 320	0.044679
2	18.0	8	318.0	150.0	3436	11.0	70	1	plymouth satellite	0.043655
3	16.0	8	304.0	150.0	3433	12.0	70	1	amc rebel sst	0.043694

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
df.groupby('origin')['mpg'].mean() #Group the cars by origin and calculate the mean mpg for each group
```

origin	mpg
1	20.083534
2	27.891429
3	30.450633

dtype: float64

```
mean_mpg_by_origin = print(df.groupby('origin')['mpg'].mean())
```

```
origin
1    20.083534
2    27.891429
3    30.450633
Name: mpg, dtype: float64
```

```
top_10_mpg_cars = print(df.sort_values(by='mpg', ascending=False).head(10)) #Sort the DataFrame by mpg in descending order and display the t
```

```
      mpg  cylinders  displacement  horsepower  weight  acceleration  \
322  46.6          4          86.0        65.0    2110          17.9
329  44.6          4          91.0        67.0    1850          13.8
325  44.3          4          90.0        48.0    2085          21.7
394  44.0          4          97.0        52.0    2130          24.6
326  43.4          4          90.0        48.0    2335          23.7
244  43.1          4          90.0        48.0    1985          21.5
309  41.5          4          98.0        76.0    2144          14.7
330  40.9          4          85.0        93.5    1835          17.3
324  40.8          4          85.0        65.0    2110          19.2
247  39.4          4          85.0        70.0    2070          18.6

     model year  origin                        car name  \
322          80       3                       mazda glc
329          80       3              honda civic 1500 gl
325          80       2             vw rabbit c (diesel)
394          82       2                       vw pickup
326          80       2                vw dasher (diesel)
244          78       2  volkswagen rabbit custom diesel
309          80       2                       vw rabbit
330          80       2             renault lecar deluxe
324          80       3                      datsun 210
247          78       3                   datsun b210 gx

     power_to_weight_ratio
322               0.030806
329               0.036216
325               0.023022
394               0.024413
326               0.020557
244               0.024181
309               0.035448
330               0.050954
324               0.030806
247               0.033816
```

```
def performance_score(row):
    return row['mpg'] * row['acceleration'] / row['weight']
```

```
df['performance_score'] = df.apply(performance_score, axis=1) #Apply this function to each row and store the result in the new column
```

```
df.head()
```

	mpg	cylinders	displacement	horsepower	weight	acceleration	model year	origin	car name	power_to_weight_ratio	performance_score
0	18.0	8	307.0	130.0	3504	12.0	70	1	chevrolet chevelle malibu	0.037100	0.061644
1	15.0	8	350.0	165.0	3693	11.5	70	1	buick skylark 320	0.044679	0.046710

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
summary_df = print(df.groupby('model year')[['mpg', 'weight', 'horsepower']].mean()) #Generate a summary DataFrame with:Average mpg, weight,
```

```
                 mpg       weight  horsepower
model year
70          17.689655  3372.793103  147.827586
71          21.250000  2995.428571  106.553571
72          18.714286  3237.714286  120.178571
73          17.100000  3419.025000  130.475000
74          22.703704  2877.925926   94.203704
75          20.266667  3176.800000  101.066667
76          21.573529  3078.735294  101.117647
77          23.375000  2997.357143  105.071429
78          24.061111  2861.805556   99.694444
79          25.093103  3055.344828  101.206897
80          33.696552  2436.655172   78.586207
81          30.334483  2522.931034   81.465517
82          31.709677  2453.548387   81.854839
```

```
df = pd.read_csv('auto-mpg.csv')
```

```
df
```

	mpg	cylinders	displacement	horsepower	weight	acceleration	model year	origin	car name
0	18.0	8	307.0	130	3504	12.0	70	1	chevrolet chevelle malibu
1	15.0	8	350.0	165	3693	11.5	70	1	buick skylark 320
2	18.0	8	318.0	150	3436	11.0	70	1	plymouth satellite
3	16.0	8	304.0	150	3433	12.0	70	1	amc rebel sst
4	17.0	8	302.0	140	3449	10.5	70	1	ford torino
...	...	...	...	...	...	...	...	...	...
393	27.0	4	140.0	86	2790	15.6	82	1	ford mustang gl
394	44.0	4	97.0	52	2130	24.6	82	2	vw pickup
395	32.0	4	135.0	84	2295	11.6	82	1	dodge rampage
396	28.0	4	120.0	79	2625	18.6	82	1	ford ranger
397	31.0	4	119.0	82	2720	19.4	82	1	chevy s-10

398 rows × 9 columns

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
subset_df = df[df["mpg"] > 30][["mpg", "cylinders", "horsepower", "weight"]] #Save a subset of the data containing only mpg, cylinders, hors
```

```
subset_df.to_csv("high_mpg_cars.csv", index=False)
```

```
Q1 = df["mpg"].quantile(0.25)  #iqr of mpg
Q3 = df["mpg"].quantile(0.75)
IQR = Q3 - Q1
```

```
print("IQR for mpg:", IQR)
```

```
IQR for mpg: 11.5
```

```
lower_bound = Q1 - 1.5 * IQR #Define outliers as values less than Q1 - 1.5 * IQR or greater than Q3 +1.5 * IQR.
upper_bound = Q3 + 1.5 * IQR


outliers = df[(df["mpg"] < lower_bound) | (df["mpg"] > upper_bound)]


print("Outliers in mpg column:")
print(outliers)
```

```
Outliers in mpg column:
      mpg  cylinders  displacement horsepower  weight  acceleration  \
322  46.6          4          86.0         65    2110          17.9

     model year  origin   car name
322          80       3  mazda glc
```

```
print("Outlier Cars:")
print(outliers)
```

```
Outlier Cars:
      mpg  cylinders  displacement horsepower  weight  acceleration  \
322  46.6          4          86.0         65    2110          17.9

     model year  origin   car name
322          80       3  mazda glc
```